# Graphes _ Parcours en Largeur

## Principe de l'algorithme de parcours en largeur ( *Breadth-First Search)*

Un site web peut être modélisé par un graphe dans lequel les sommets représentent les pages du site et les arêtes représentent les liens hypertextes permettant d'aller d'une page à une autre.

On suppose que ce graphe est connexe, c'est-à-dire qu'à partir d'une page donnée, toute autre page est accessible par une série de liens (une chaîne).

On cherche à tester toutes les pages de ce site, c'est-à-dire à parcourir ce graphe en passant par tous ses sommets.

On va procéder à un parcours en largeur du graphe en mettant les sommets successifs dans une file (structure FIFO).

## Description intuitive de l'algorithme :

1. On enfile le sommet de départ (on visite la page d'accueil du site).
2. On enfile les sommets adjacents à la tête de file (on visite les pages ciblées par la page d'accueil) s'ils ne sont pas déjà présents dans la file.
3. On défile (c'est-à-dire on supprime la tête de file).
4. Tant que la file n'est pas vide, on réitère les points 2 et 3.

En d'autres termes, on défile toujours prioritairement les sommets (les pages) les plus tôt découverts.

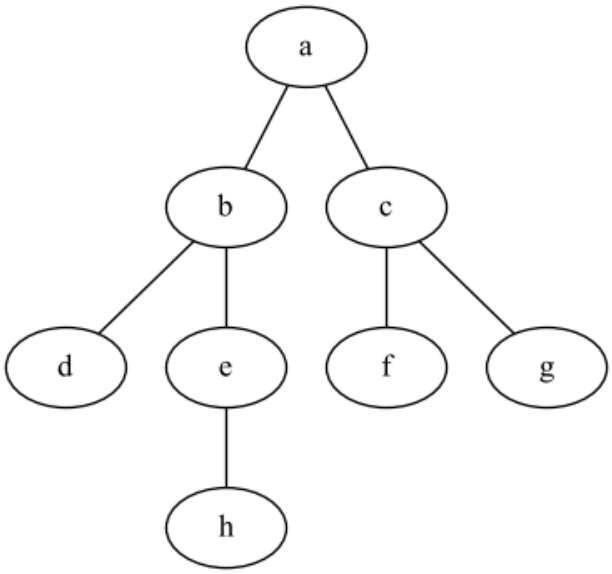
## En pseudo code

```
ParcoursLargeur(Graphe G, Sommet s):
        f = CreerFile();
        f.enfiler(s);
        marquer(s);
        tant que la file est non vide
                s = f.defiler();
                afficher(s);
                pour tout voisin t de s dans G
                    si t non marqué
                            f.enfiler(t);
                            marquer(t);
```

# Graphes _ Parcours en Largeur

**Déroulement à la main**

**Graphe G**




---

| **On applique** `ParcoursLargeur(G, a):` | | | |
|---|---|---|---|
| `f = CreerFile();` | **file** | **marque** | **afficher** |
| `f.enfiler(s);`<br>`marquer(s);` | **a** | **a** | |
| `f.defiler()`<br>`afficher(s)` | | **a** | **a** |
| `f.enfiler(t);`<br>`marquer(t);` | **cb** | **cba** | |
| | | | |
| `f.defiler()`<br>`afficher(s)` | **c** | | **b** |
| | **edc** | **edcba** | |
| | **ed** | **gfed** | **c** |

# Graphes _ Parcours en Largeur

**Q 1 Insérer la classe File à votre programme qui possède déjà une classe Graphe (pour les graphes non pondérés )**

**Q2 Définir un dictionnaire pour représenter le graphe en exemple**

**Q3 Définir la méthode Parcours en Largeur "bfs" Breadth-First Search**

Cette méthode commence à parcourir le graphe à partir du premier sommet. Le code permettant de récupérer du dictionnaire de définition est donné ci-après.

```
def bfs(self):
        #Obtention du premier couple clé/valeur du dictionnaire
        els = list(self.dic.items())
        #valeur du premier sommet
        s_premier = els[0][0]

        #Marquage du premier sommet
        Marque= [s_premier]
```

Utiliser le speudo code pour finir cette méthode.

Vous devez obtenir pour l'exemple choisi : ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']